Parallelization of Implicit-Explicit Runge-Kutta Methods for Cluster of PCs

José Miguel Mantas[1], Pedro González[2], and José A. Carrillo[3]

[1] Software Engineering Department. University of Granada
C/ P. Daniel de Saucedo s/n. E-18071 Granada, Spain
jmmantas@ugr.es

[2] Department of Applied Mathematics. University of Granada
Avda. Fuentenueva s/n. E-18071 Granada, Spain
prodelas@ugr.es

[3] Departament de Matemàtiques - ICREA. Universitat Autònoma de Barcelona
Bellaterra E-08193
carrillo@mat.uab.es

Abstract. Several physical phenomena of great importance in science and engineering are described by large partly stiff differential systems where the stiff terms can be easily separated from the remaining terms. Implicit-Explicit Runge-Kutta (IMEXRK) methods have proven to be useful solving these systems efficiently. However, the application of these methods still requires a large computational effort and their parallel implementation constitutes a suitable way to achieve acceptable response times. In this paper, a technique to parallelize and implement efficiently IMEXRK methods on PC clusters is proposed. This technique has been used to parallelize a particular IMEXRK method and an efficient parallel implementation of the resultant scheme has been derived in a structured manner by following a component-based approach. Several numerical experiments which have been performed on a cluster of dual PCs reveal the good speedup and the satisfactory scalability of the parallel solver obtained.

1 Introduction

The spatial discretization of a great variety of time-dependent partial differential equations (PDEs) by the method of lines leads to large systems of ordinary differential equations (ODEs) with this form:

$$\frac{dy}{dt} = \mathbf{f}(y) + \mathbf{g}(y), \qquad y(0) = y_0 \in \mathbb{R}^d, \qquad t > 0 \tag{1}$$

where $y = y(t) \in \mathbb{R}^d$ is the unknown function of a d-dimensional ODE system which is defined by the component functions $\mathbf{f}, \mathbf{g} : \mathbb{R}^d \longrightarrow \mathbb{R}^d$. The function $\mathbf{g}(y)$ results from the discretization of the stiff terms and $\mathbf{f}(y)$ results from the discretization of the remaining terms. The function $\mathbf{g}$ is usually written as $(1/\epsilon)\tilde{\mathbf{g}}$ ($\tilde{\mathbf{g}} : \mathbb{R}^d \longrightarrow \mathbb{R}^d$), where $\epsilon > 0$ is the stiffness parameter [5].

There are many practical problems where it can be advantageous to integrate $\mathbf{f}$ explicitly, to reduce computational costs, and $\mathbf{g}$ implicitly, to avoid excessively small time steps. In fact, the Jacobian of $\mathbf{g}$ in problems of form (1) frequently exhibits a particular structure (positive definite, symmetric, and sparse) whose exploitation would make it possible a considerable saving of computational effort if $\mathbf{g}$ is implicitly integrated. This special structure can be lost if global implicit methods are used to integrate both terms ($\mathbf{f}(y)+\mathbf{g}(y)$). Therefore, it often makes sense to integrate $\mathbf{g}(y)$ implicitly and $\mathbf{f}(y)$ explicitly in these problems when they exhibit this structure.

A clear example of this type of system appears in reaction-diffusion and convection-diffusion problems [1, 7, 12, 13] arising in multiple areas of science and engineering. In these problems, an explicit scheme would be used for the reaction (resp. convection) term and an implicit scheme for the diffusion term.

There exists Runge-Kutta methods which are specially suitable for systems of form (1). These schemes, known as *Implicit-Explicit Runge-Kutta Methods (IMEXRK)*, apply an implicit discretization for $\mathbf{g}$ and an explicit one for $\mathbf{f}$, simultaneously, in the same time step and using identical time step size. A Diagonally Implicit Runge-Kutta method (DIRK) [2] is usually considered to integrate $\mathbf{g}$, given the importance of the efficiency in the solution of the stiff part of the equation [1].

However, the application of an IMEXRK method, together with the complexity of these systems, demands a great deal of computing power which can be easily achieved by using efficient parallel implementations running on cluster of Personal Computers (PCs). In this paper, the development of parallel software based on IMEXRK methods is tackled.

In section 2, the structure of the most relevant IMEXRK methods is briefly presented. A general technique to parallelize these numerical methods will be described in section 3. This technique will be applied to a particular IMEXRK method in section 4, to obtain a new parallel scheme. A component based methodological approach for deriving group parallel ODE solvers [9, 10] is used in section 5 to develop an efficient parallel implementation of this numerical scheme. This approach enables the exploitation of the multilevel parallelism which exhibits the numerical scheme and the ODE system in an structured manner. This implementation is adapted to solve a 1D rarefied gas shock profile on a cluster of dual PCs. Section 6 presents the experimental results obtained with the parallel solver. Finally, section 7 gives the main conclusions of the work.

2 Implicit-Explicit Runge-Kutta Methods

We have considered IMEXRK methods where the implicit solver (which is used to integrate $\mathbf{g}$) is a DIRK method. With this requirement, we have identified two relevant types of IMEXRK methods: pure IMEXRK methods [1, 8] and Additive Semi-Implicit Runge-Kutta methods of type A (ASIRK-A) [13].

A pure s-stage IMEXRK scheme is characterized by two matrices $\tilde{A}, A \in \mathbb{R}^{s\times s}$ ($\tilde{A} = (\tilde{a}_{ij})$, $A = (a_{ij})$) and two coefficient vectors $\tilde{b}, b \in \mathbb{R}^s$ ($\tilde{b} =$

$(\tilde{b}_1, \ldots, \tilde{b}_s)$, $b = (b_1, \ldots, b_s)$). The matrix $\tilde{A}$ is strictly lower triangular ($\tilde{a}_{ij} = 0$, for $j \geq i$) and A is lower triangular ($\tilde{a}_{ij} = 0$, for $j > i$). When these parameters are applied to a system of form (1), we obtain the numerical scheme which appears below. This scheme describes how to obtain the vector $y_n \in \mathbb{R}^d$, which approximates $y(t_n)$, from the approximation given by the previous integration step $y_{n-1} \approx y(t_{n-1})$:

$$\begin{aligned} Y_{n,i} &= y_{n-1} + h_n \left(\sum_{j=1}^{i-1} \tilde{a}_{ij} \mathbf{f}(Y_{n,j}) + \sum_{j=1}^{i} a_{ij} \mathbf{g}(Y_{n,j}) \right), \qquad i = 1, \ldots, s \\ y_n &= y_{n-1} + h_n \left(\sum_{i=1}^{s} \tilde{b}_i \mathbf{f}(Y_{n,i}) + \sum_{i=1}^{s} b_i \mathbf{g}(Y_{n,i}) \right), \qquad n = 1, \ldots, N_{steps} \end{aligned} \tag{2}$$

where $Y_{n,i} \in \mathbb{R}^d, \quad i = 1, \ldots, s$ and h_n is the *size* of the n-th time step ($h_n = t_n - t_{n-1}$). N_{steps} is the total number of time steps.

An s-stage ASIRK-A scheme (ASIRK-sA) is represented with the same kind of parameters as an s-stage IMEXRK method. The only difference is that in ASIRK-A methods, only a coefficient vector $b \in \mathbb{R}^s$ is necessary, in addition to the matrices $\tilde{A}, A \in \mathbb{R}^{s \times s}$ which maintain the same previously mentioned structure. When these parameters are applied to a system of form (1), we obtain:

$$\begin{aligned} Y_{n,i} &= h_n \left[\mathbf{f}\left(y_{n-1} + \sum_{j=1}^{i-1} \tilde{a}_{ij} Y_{n,j} \right) + \mathbf{g}\left(y_{n-1} + \sum_{j=1}^{i} a_{ij} Y_{n,j} \right) \right], i = 1, .., s \\ y_n &= y_{n-1} + \sum_{i=1}^{s} b_i Y_{n,i}, \qquad n = 1, \ldots, N_{steps} \end{aligned} \tag{3}$$

3 An Approach to Parallelize IMEXRK Methods

When an IMEXRK method is applied to a system of form (1), s connected d-dimensional nonlinear systems must be solved sequentially (see (2) and (3)).

If the modified Newton method [2] is applied to solve each of these systems, we obtain s Newton iterations where the i-th iteration ($i = 1, \ldots, s$) computes an approximation to the vector $Y_{n,i}$. Each iteration must be solved before the next one, because the i-th iteration depends on vectors $Y_{n,j}, \quad j = 1, \ldots, i-1$, which must be computed in previous iterations. As a result, the following numerical scheme is obtained:

```
for $n = 1, \ldots, N_{steps}$
    for $i = 1, \ldots, s$
        $Y_{n,i}^{(0)}$ is computed by using a predictor formula
        for $v = 1, \ldots, m_i$
            $R_i = Q_i - Y_{n,i}^{(v-1)}$
            $Y_{n,i}^{(v)} = Y_{n,i}^{(v-1)} + \left( I_d - a_{ii} h_n \mathbf{J}_{\mathbf{g}}(k_i) \right)^{-1} R_i$
```

The values m_i of this method are dynamically determined to ensure that $Y_{n,i}^{(m_i)}$ is a good approximation of $Y_{n,i}$. $\mathbf{J}_{\mathbf{g}}(k_i)$ denotes an approximation to the Jacobian of $\mathbf{g}$ evaluated at k_i and I_d denotes the d-dimensional identity matrix.

The values of Q_i and k_i vary depending on the type of IMEXRK scheme which is considered. So, for pure IMEXRK schemes, $k_i = y_{n-1}$ and

$$Q_i = y_{n-1} + h_n \left(\sum_{j=1}^{i-1} \tilde{a}_{ij} \mathbf{f}(Y_{n,j}) + \sum_{j=1}^{i-1} a_{ij} \mathbf{g}(Y_{n,j}) + a_{ii} \mathbf{g}(Y_{n,i}^{(v-1)}) \right),$$

and for ASIRK-A schemes, $k_i = y_{n-1} + \sum_{j=1}^{i-1} a_{ij} Y_{n,j}$ and

$$Q_i = h_n \left[\mathbf{f}\left(y_{n-1} + \sum_{j=1}^{i-1} \tilde{a}_{ij} Y_{n,j} \right) + \mathbf{g}\left(y_{n-1} + \sum_{j=1}^{i-1} a_{ij} Y_{n,j} + a_{ii} Y_{n,i}^{(v-1)} \right) \right].$$

Since the iterations must be solved sequentially, this scheme does not exhibit a lot of task parallelism exploitable across the method [2]. In order to decouple the calculations associated to each stage, we propose to approximate the solution of this scheme by using a similar scheme in which the calculations of each stage can be performed in parallel. The new scheme introduces some of redundant computation, but this additional cost is relatively small. This scheme is based on the following reasonable assumptions:

- For ASIRK-A schemes, the Jacobian of $\mathbf{g}$ in $y_{n-1} + \sum_{j=1}^{i-1} a_{ij} Y_{n,j}$ is approximated by the Jacobian evaluated in y_{n-1} ($k_i = y_{n-1}$). This approximation does not involve a considerable loss of accuracy and makes it possible to use the same Jacobian matrix in the calculations of all the stages.
- The calculations of each stage can be performed in a synchronous and concurrent manner if, in the v-th iteration of the original scheme, we approximate $Y_{n,j}$ by $Y_{n,j}^{(v-1)}$ and we consider the same number of Newton iterations for all the stages ($v = 1, \ldots, m$).

Now, the parallel numerical schemes which result from these assumptions are described. In these descriptions, the term $Y_{n,j}^{(v-1)}$ appears highlighted to emphasize the modification of the original schemes.

```
for $n = 1, \ldots, N_{steps}$
    $Y_n^{(0)} = Pred(Y_{n-1})$        ($Pred(\cdot)$ denotes a predictor formula)
    for $v = 1, \ldots, m$
        parfor $i = 1, \ldots, s$
            $R_i = Q_i - Y_{n,i}^{(v-1)}$
            $Y_{n,i}^{(v)} = Y_{n,i}^{(v-1)} + \left( I_d - a_{ii} h_n \mathbf{J}_{\mathbf{g}}(y_{n-1}) \right)^{-1} R_i$
```

where $Y_n \in \mathbb{R}^{sd}$ is the so-called *stage vector*, which contains s d-dimensional components $Y_{n,i}, \quad i = 1, \ldots, s$ $(Y_n = (Y_{n,1}, \ldots, Y_{n,s})^T)$. The value of m must be dynamically determined to ensure that $(Y_{n,1}^{(m)}, \ldots, Y_{n,s}^{(m)})^T$ is a good approximation of Y_n. Here the term Q_i is different. For IMEXRK schemes:

$$Q_i = y_{n-1} + h_n\left(\sum_{j=1}^{i-1} \tilde{a}_{ij}\mathbf{f}(\boxed{Y_{n,j}^{(v-1)}}) + \sum_{j=1}^{i-1} a_{ij}\mathbf{g}(\boxed{Y_{n,j}^{(v-1)}}) + a_{ii}\mathbf{g}(Y_{n,i}^{(v-1)})\right)$$

and for ASIRK-A schemes:

$$Q_i = h_n\left[\mathbf{f}\left(y_{n-1} + \sum_{j=1}^{i-1} \tilde{a}_{ij}\boxed{Y_{n,j}^{(v-1)}}\right) + \mathbf{g}\left(y_{n-1} + \sum_{j=1}^{i-1} a_{ij}\boxed{Y_{n,j}^{(v-1)}} + a_{ii}Y_{n,i}^{(v-1)}\right)\right]$$

With this scheme, the total number of Newton iterations would be no less than in the original scheme $(m \geq \max_{1\leq i\leq s}(m_i))$. However, this additional number of iterations does not involve an excessive loss of efficiency.

4 Application to a Particular IMEXRK Method

Now the derivation of a particular parallel IMEXRK method of second order is performed. The selected method is termed LRR(3,2,2) [8] (3 stages are used in the implicit scheme, 2 stages in the explicit scheme and the convergence order is 2). This method is very suitable in applications where the stiff terms can be easily separated from the rest of the equations and a high accuracy is not required in the time discretization [1]. The parameters which characterize this method [8] lead to the following numerical scheme:

$$Y_{n,1} = y_{n-1} + \frac{1}{2}h_n[\mathbf{f}(y_{n-1}) + \mathbf{g}(Y_{n,1})], \qquad Y_{n,2} = y_{n-1} + \frac{1}{3}h_n[\mathbf{f}(y_{n-1}) + \mathbf{g}(Y_{n,2})]$$

$$y_n = Y_{n,3} = y_{n-1} + h_n[\mathbf{f}(Y_{n,1}) + \frac{3}{4}\mathbf{g}(Y_{n,2}) + \frac{1}{4}\mathbf{g}(Y_{n,3})] \tag{4}$$

This scheme exhibits exploitable task parallelism itself, because the computation of vectors $Y_{n,1}$ and $Y_{n,2}$ can be performed simultaneously. However, the computation of $y_n = Y_{n,3}$ requires the previous computation of those two vectors. We have applied the previously described general technique to enable a higher degree of concurrency, obtaining the following parallel scheme, which has been termed *PIMEXRK3* (Parallel IMEXRK method with 3 stages).

```
for $n = 1, \ldots, N_{steps}$ {    $Y_n^{(0)} = Pred(Y_{n-1})$    // $a_{11} = \frac{1}{2},\ a_{22} = \frac{1}{3},\ a_{33} = \frac{1}{4}$
  for $v = 1, \ldots, m$ {
    par { parfor $i = 1, 2$ {$R_i = y_{n-1} + a_{ii}h_n\left[\mathbf{f}(y_{n-1}) + \mathbf{g}(Y_{n,i}^{(v-1)})\right] - Y_{n,i}^{(v-1)}$}

          $R_3 = y_{n-1} + h_n\left[\mathbf{f}(Y_{n,1}^{(v-1)}) + \frac{3}{4}\mathbf{g}(Y_{n,2}^{(v-1)}) + a_{33}\mathbf{g}(Y_{n,3}^{(v-1)})\right] - Y_{n,3}^{(v-1)}$}

    parfor $i = 1, 2, 3$ { $Y_{n,i}^{(v)} = Y_{n,i}^{(v-1)} + (I_d - a_{ii}h_n\mathbf{J_g}(y_{n-1}))^{-1}R_i$ } }
}
```

5 Derivation of Parallel Implementations of the PIMEXRK3 Scheme

Following a component-based approach to derive parallel ODE solvers [9, 10], termed COMPODES, a distributed implementation of the PIMEXRK3 scheme has been obtained to solve a particular problem on a cluster of dual PCs, by appplying 3 phases in sequence.

1. Component-Based Generic Description of the Numerical Scheme From the mathematical description of the numerical scheme, the first phase of COMPODES is applied. For that purpose, several abstract operations are selected and combined suitably to describe the algorithm and to express the maximum degree of task parallelism. A summarized generic description of the PIMEXRK3 method, based on abstract operations, is shown in Figure 1a), where the edges denote data dependencies and the main sources of task parallelism are represented with concurrent loops ($PAR\ i = 1, 3$). We have selected a direct method based on LU decomposition to solve the linear systems, because this choice enables the reuse of the same LU decomposition for the solution of all the linear systems which arise in one time step. In fact, The operation `LUdecomp`$(.., A, ..)$ denotes the LU factorization of A, `SolveSystem`$(.., A, ..., X)$ denotes the computation of $X \longleftarrow A^{-1}X$ (assuming `LUdecomp`$(.., A, ..)$) and `Feval`$(.., t, f, y, dy)$ denotes the evaluation of a function f.

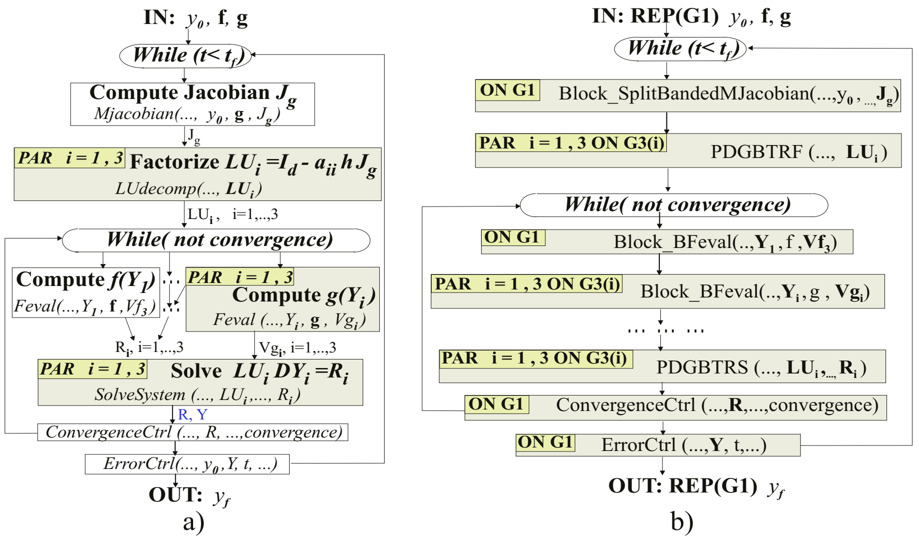

Fig. 1. a) Generic description of PIMEXRK3, b) Description of task scheduling

2. Adaptation to a Particular ODE System The generic description of the PIMEXRK3 method has been adapted, following the second COMPODES phase, to perform the time integration of a hydrodynamical model of the Boltzmann equation for rarefied gases in 1D [11]. This model is based on a system of 5 PDEs, termed the relaxed Burnett system [6], which can be written in the following form:

$$\begin{pmatrix} U_t \\ V_t \end{pmatrix} = \begin{pmatrix} -F(U,V)_x \\ -G(U,V,U_x,V_x) + D(U,V,U_x,V_x)_x \end{pmatrix}, \text{ where} \tag{5}$$

$$U = \begin{pmatrix} \rho \\ m = \rho u \\ z = \frac{1}{2}\rho u^2 + \frac{3}{2}p \end{pmatrix}, \quad V = \begin{pmatrix} \sigma \\ q \end{pmatrix}, \quad F(U,V) = \begin{pmatrix} \rho u \\ \rho u^2 + p + \sigma \\ \frac{1}{2}\rho u^3 + \frac{5}{2}up + \sigma u + q \end{pmatrix}.$$

The terms $G(U,V,U_x,V_x)$ and $D(U,V,U_x,V_x)$ are defined in [6]. In this system, we have five independent variables ρ (mass density), $m = \rho u$ (momentum, where u is the macroscopic velocity), $z = \frac{1}{2}\rho u^2 + \frac{3}{2}p$ (total energy, where p is the normal pressure), σ (pressure deviation tensor) and q (heat flux vector).

A spatial discretization of this system is proposed in [11]. This discretization is based on combining relaxation schemes for the conservative part and standard second order central differences for the non conservative part. The resulting system has 5N ODEs when the 1D space is discretized by using N grid points. A suitable arrangement of the equations leads to a a narrow banded ODE system whose Jacobian matrix has 9 subdiagonals and 7 superdiagonals.

The stiff and nonstiff terms in (5) have the following form:

$$\mathbf{f}(U,V) = (-F(U,V)_x, 0)^T, \quad \mathbf{g}(U,V) = (0, D(U,V,U_x,V_x)_x - G(U,V,U_x,V_x))^T.$$

If we maintain the same spatial discretization and arrangement which it is proposed in [11], the subsystem associated to $\mathbf{g}$ has a banded structure and the Jacobian of $\mathbf{g}$ has 9 subdiagonals and 5 superdiagonals. Therefore, the banded structure of the function $\mathbf{g}$ and its Jacobian is narrower than in the original system ($f = \mathbf{f} + \mathbf{g}$). Since the structure of the Jacobian matrix for $\mathbf{g}$ determines the complexity of the more costly calculations in the implicit time integration, the use of an IMEXRK method involves an important reduction of computational costs.

The initial vector y_0 of the ODE system captures the state before a one-dimensional shock profile with Mach number 10 [11].

In order to enable the exploitation of the particular structure of the subsystems $\mathbf{f}$ and $\mathbf{g}$, the generic specification of the PIMEXRK3 scheme has been adapted by replacing several operations (for instance, `LUdecomp`, `MJacobian` and `Feval`) by specializations which assume a banded structure [10].

3. Parallel Design Decisions

Following the third COMPODES phase, several parallel design decisions have been made in order to compute efficiently the specialized PIMEXRK3 scheme on a processor number, P, which is multiple of 3. These decisions include the scheduling of the tasks and the selection of the best data parallel implementation and data distribution to realize each operation. An approximation method has been proposed to make these decisions systematically [10].

A summarized graphical description of some of these decisions is shown in Figure 1b). Several groups of processors have been considered to schedule the tasks: a global group with P processors (**G1**), and 3 disjointed subgroup with $P/3$ processors (**G3(i)**, i=1,2,3). To compute an approximation of the Jacobian of $\mathbf{g}$, we chose an optimal implementation for banded Jacobians `Block_SplitBandedMJacobian` on the global group **G1**. This implementation generates a block column distribution of a compact representation of the Jacobian. The evaluations of the function $\mathbf{f}$ are performed on the group **G1** while the evaluations of $\mathbf{g}$ for the i-th stage ($i = 1, 2, 3$) are performed on **G3(i)**. These evaluations are implemented by using a parallel block routine (`Block_BFeval`) which takes into account the banded structure of the ODE system terms in order to reduce the remote communication. The LU decompositions and system solutions for the i-th stage are computed on **G3(i)** by using the routines of the ScaLAPACK library [3] `PDGBTRF` (banded LU Factorization) and `PDGBTRS` (banded system solution). These routines takes advantage of the banded structure of the system and follows a block column distribution.

These decisions have been translated into a parallel program which is expressed in Fortran augmented with routines of ScaLAPACK and *MPI* [4].

6 Numerical Experiments

We have performed several numerical experiments on a cluster of 8 dual AMD processors 2.5Ghz, running Linux, connected via a Gigabit ethernet switch.

Table 1. Comparison among LRR(3,2,2) and PIMEXRK3 numerical solutions

$h_n=$	0.5	0.25	0.125	0.0625
$\|\|y_{LRR} - y_{P3}\|\|_2$	$5.599 \cdot 10^{-2}$	$7.956 \cdot 10^{-5}$	$2.223 \cdot 10^{-5}$	$5.970 \cdot 10^{-6}$

In order to show the accuracy of the numerical results obtained with the parallel solver, we compare the numerical solutions obtained with a sequential implementation of the LRR(3,2,2) method (y_{LRR}) and with an implementation of the PIMEXRK3 scheme running on 6 processors (y_{P3}). Table 1 shows the L^2-norm of the difference between the numerical solutions obtained with both solvers ($||y_{LRR} - y_{P3}||_2$). These results have been obtained for $t = 5.0$ and $N = 200$. Several experiments has been performed with a fixed step size although a different step size has been used in each experiment. The results prove that there is a great agreement between the solutions obtained by both methods.

The time results (in seconds) obtained for different values of N on several processor numbers P are shown in Table 2. The speedup results are graphically shown in Figure 2. These speedup results have been obtained by comparing the parallel execution time for several values of P, with the execution time of a sequential implementation of the $LRR(3, 2, 2)$ scheme running on a single

Table 2. Time results for the PIMEXRK3 scheme applied to the test problem

N =	72	396	900	1296	1800	2520	3600	3996	7200	9000
LRR($P = 1$)	0.258	1.417	3.222	4.644	6.451	9.026	12.94	14.37	25.89	32.37
$P = 3$	0.0899	0.476	1.093	1.585	2.1932	3.074	4.383	4.867	8.76	11.76
$P = 6$	0.0559	0.259	0.572	0.824	1.1439	1.591	2.292	2.527	4.55	5.71
$P = 9$	0.063	0.227	0.407	0.569	0.791	1.108	1.562	1.756	3.09	3.78
$P = 12$	0.0424	0.153	0.311	0.435	0.6013	0.822	1.173	1.286	2.32	2.87
$P = 15$	0.1906	0.273	0.298	0.399	0.5167	0.707	0.970	1.067	1.94	2.36

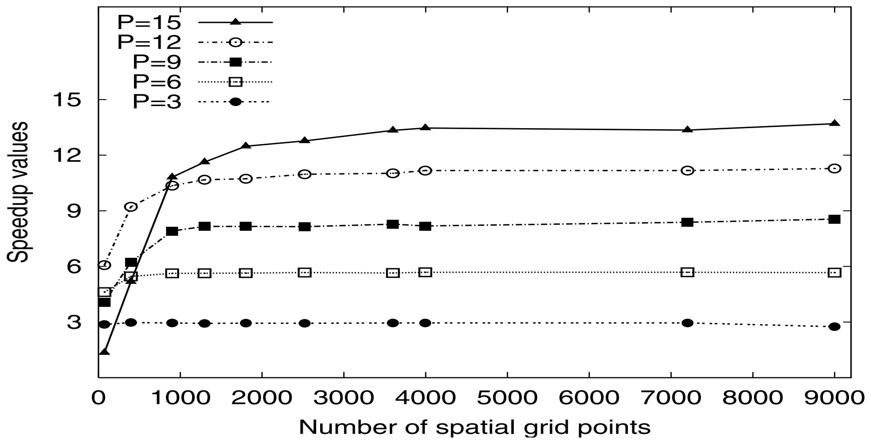

Fig. 2. Speedup results for several spatial grid sizes

processor. This implementation of the $LRR(3,2,2)$ scheme also takes advantage of the banded structure of the subsystems. The time results were taken for one time step and meshes varying from $N = 80$ to $N = 4000$ grid points.

The experiments show that a speedup close to the linear speedup can be achieved when N is greater than 500. In general, efficiencies higher than 90% are achieved with a sufficiently large number of grid points, except on $P = 15$ processors where a larger problem size would be necessary to achieve this efficiency. The results reveal the parallelism which exhibits the new parallel numerical scheme and the satisfactory scalability which offers the implementation derived by the COMPODES approach.

7 Conclusions

A method to obtain efficient implementations of IMEXRK methods for PC clusters has been described. This method incorporates two procedures:

1. A technique to decouple the computation associated with each stage, when an s-stage IMEXRK method is applied to a partly stiff system with d ODEs,

has been introduced. The technique is based on considering the s coupled d-dimensional nonlinear systems, which arise when an IMEXRK method is applied, as only one sd-dimensional nonlinear system, and imposing several reasonable approximation assumptions when the modified Newton method is used to solve the nonlinear system. As a result, the calculations associated with each stage of the method can be performed in parallel for each round of the Newton iteration.
2. A component based approach [10] can be applied to easily derive efficient implementations of the previously defined parallel schemes. This approach allows us to exploit the task and data parallelism which exhibits the scheme by using software components of parallel libraries.

The method has been illustrated by deriving an efficient implementation of a second order 3-stage IMEXRK method. The experimental results obtained on a cluster of dual PCs reveal the good speedup and the satisfactory scalability of the parallel solver for the range of processor numbers which has been considered.

Acknowledgements

The authors acknowledge support from the European IHP network HYKE "Hyperbolic and Kinetic Equations: Asymptotics, Numerics, Applications" HPRN-CT-2002-00282. JM and JAC acknowledge partial support from DGI-MCYT/FEDER project BFM2002-01710. PG acknowledges partial support from DGI-MCYT/FEDER project BFM2002-02649. JM also acknowledges partial support from MEC/FEDER project TIN2004-07672-c03-02.

References

1. Ascher, U. M., Ruuth, S. J., Spiteri, R. J.: Implicit-Explicit Runge-Kutta Methods for time-dependent Partial Differential Equations. Applied Numerical Mathematics. **25** (1997) 151-167
2. Burrage, K.: Parallel and Sequential Methods for Ordinary Differential Equations. Oxford Science Publications. (1995)
3. Dongarra, J., Walker, D. W.: Software libraries for linear Algebra Computations on High Performance Computers. SIAM Review. **37** (1995) 151-180
4. Message Passing Interface Forum. MPI: A Message Passing Interface Standard. Univ. of Tennessee, Knoxville, Tennessee, (1995)
5. Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II: Stiff and Differential Algebraic Problems. Springer-Verlag. (1996).
6. Jin, S., Pareschi, L., Slemrod, M.: A Relaxation Scheme for Solving the Boltzmann Equation Based on the Chapman-Enskog Expansion. Acta Mathematicas Applicatae Sinica (English Series). **18** (2002) 37-62
7. Kennedy, C. A., Carpenter, M. H.: Additive Runge-Kutta schemes for convection-diffusion-reaction equations. Applied Numerical Mathematics. **1** (2003) 139-181
8. Pareschi, L., Russo, G.: Implicit-Explicit Runge-Kutta schemes for stiff systems of differential equations. In Recent Trends in Numerical Analysis. **3** (2000) 269-289

9. Mantas, J. M., Ortega, J., Carrillo, J. A.: Component-Based Derivation of a Stiff ODE Solver implemented on a PC Cluster. International Journal of Parallel Programming. **30** (2002) 99-148
10. Mantas, J. M., Ortega, J., Carrillo, J. A.: Integrating Multiple Implementations and Structure Exploitation in the Component-based Design of Parallel ODE Solvers. Recent Advances in Parallel Virtual Machine and Message-Passing Interface. Lecture Notes in Computer Science. **2840** (2003) 438-446
11. Mantas, J. M. , Pareschi, L., Carrillo, J. A., Ortega, J.: Parallel Integration of Hydrodynamical Approximations of the Boltzmann Equation for rarefied gases on a Cluster of Computers. J. Comp. Methods in Science and Engineering. **4** (2004) 33-41
12. Verwer, J.G., Sommeijer, B. : An implicit-explicit Runge-Kutta-Chebyshev scheme for diffusion-reaction equations. SIAM J. of Sci. Comp. **25** (2004) 1824-1835
13. Zhong, X.: Additive Semi-Implicit Runge-Kutta Methods for Computing High-Speed Nonequilibrium Reactive Flows. Journal of Comp. Physics. **128** (1996) 19-31